
Downloaded from http://ssudl.solent.ac.uk/1396/
An Investigation of Rule Induction Based Prediction Systems

Carolyn Mair and Martin Shepperd
Empirical Software Engineering Research Group
Bournemouth University
Talbot Campus Poole, BH12 5BB, UK
{cmair, mshepper}@bmth.ac.uk
http://dec.bmth.ac.uk/ESERG

14 February, 1999

Abstract
Traditionally, researchers have used either off-the-shelf models such as COCOMO, or developed local models using statistical techniques such as stepwise regression, to predict software effort estimates. More recently, attention has turned to a variety of machine learning methods such as artificial neural networks (ANNs), case-based reasoning (CBR) and rule induction (RI). This position paper outlines some preliminary research into the use of rule induction methods to build software cost models. We briefly describe the use of rule induction methods and then apply the technique to a dataset of 81 software projects derived from a Canadian software house in the late 1980s. We show that RI methods tend to be unstable and generally predict with quite variable accuracy. Pruning the feature set, however, has a significant impact upon accuracy. We also compare our results with a prediction system based upon a standard regression procedure. We suggest that further work is carried out to examine the effects of the relationships among, and between, the features of the attributes on the generated rules in an attempt to improve on current prediction techniques and enhance our understanding of machine learning methods.

KEYWORDS: rule induction, software cost models, software effort estimation, machine learning, prediction systems.

1. Background to Research
Every day, businesses need to decide how to allocate valuable resources based on predictions. Unfortunately whilst most practitioners recognise the importance of accurate predictions of development effort, current estimation techniques are often highly inaccurate. Traditionally, researchers have estimated software effort by means of off-the-shelf algorithmic models such as COCOMO (Boehm, 1981), where effort is expressed as a function of size; or have developed local models using statistical techniques such as stepwise regression. These models have been found to predict highly inaccurate estimates (Kemerer, 1987). More recently attention has turned to a variety of machine learning methods to predict software development effort. ANNs, CBR and RI are examples of such methods. This brief paper outlines some preliminary research into the use of RI methods to build software cost models.
RI is a particular aspect of inductive learning in which algorithms produce rules as a result of modelling.

“...algorithms for induction which given a training set of examples, each of which is described by the values of an attribute and the outcome, will automatically build decision trees that will correctly classify not only all the examples in the training set, but unknown examples from the wider universe of examples of which the training set is presumed to provide a representative sample.” (Kennedy et al., 1997, p.147)

Inductive learning is then the process of acquiring general concepts from specific examples. By analysing many examples, it may be possible to derive a general concept that defines the production conditions.

In order to produce a set of rules, induction works on a randomly, or algorithmically selected sub-set of the examples often referred to as the training set. These rules can be tested on the remainder of the examples (the validation or test set) to assess how well they represent the data. RI can be used for a range of problems where there exists a set of suitable examples. Rules can be seen as decision trees where the root node contains the predicted value. Numeric decision trees are generated by calculating the average outcome for the set of cases being considered at each node. An example fragment of rules generated from the Desharnais dataset is depicted below.

```plaintext
If AdjFPs >=266 and
   If ExpPM < and
       Transactions <165
       Year Fin >=85
   Then effort =3542
```

One advantage of inductive learning over neural network learning is that the rules are transparent and therefore can be read and understood. In the above example we see that adjusted function points is the first factor that is assessed followed by the number of transactions processed and the year of completion. Proponents of RI argue that this helps the estimator understand any prediction made by systems of this type.

2. Method
In order to explore the potential of RI techniques for building effort prediction models we used the data mining software package Clementine and applied it to a dataset of 81 software projects derived from a Canadian software house in the late 1980s (Desharnais 1989).
The dataset comprised the following features:

- Effort (measured in hours)
- ExpEquip (team experience in years)
- ExpProjMan (project manager’s experience in years)
- Trans (number of transactions processed)
- Entities (number of entities)
- RawFPs (unadjusted function points)
- AdjFPs (adjusted function points)
- DevEnv (development environment)
- YearFin (year of completion).

Four of the 81 projects contained missing values so were excluded from further investigation. The procedure adopted was to randomly partition the dataset into a training set of 67 projects and validation sets of 10 projects. This was done three times yielding validation sets 1, 2 and 3 so as to help assess the stability of any prediction systems generated. In addition, we used a least squares regression (LSR) procedure to provide a benchmark comparison, again model fitting on the same training sets and testing on the remaining 10 projects. Three accuracy indicators were employed:

- sum of the squares of the residuals (risk averse)
- percentage error (to indicate bias, if any)
- mean magnitude of relative error (MMRE) (to indicate the spread of estimation error)

3. Results

<table>
<thead>
<tr>
<th>Method</th>
<th>Validation Set</th>
<th>SSR</th>
<th>% Error</th>
<th>MMRE</th>
</tr>
</thead>
<tbody>
<tr>
<td>RI (All features)</td>
<td>1</td>
<td>5.69E+08</td>
<td>39%</td>
<td>86%</td>
</tr>
<tr>
<td>RI (All features)</td>
<td>2</td>
<td>1.55E+08</td>
<td>132%</td>
<td>140%</td>
</tr>
<tr>
<td>RI (All features)</td>
<td>3</td>
<td>2.51E+08</td>
<td>61%</td>
<td>87%</td>
</tr>
<tr>
<td>RI (Excl. DevEnv)</td>
<td>1</td>
<td>1.74E+08</td>
<td>12%</td>
<td>41%</td>
</tr>
<tr>
<td>LSR - 1</td>
<td>1</td>
<td>1.00E+08</td>
<td>27%</td>
<td>47%</td>
</tr>
<tr>
<td>LSR - 2</td>
<td>2</td>
<td>0.21E+08</td>
<td>24%</td>
<td>38%</td>
</tr>
<tr>
<td>LSR - 3</td>
<td>3</td>
<td>1.90E+08</td>
<td>-98%</td>
<td>100%</td>
</tr>
</tbody>
</table>

Table 1: Comparative accuracy of Rule Induction and Least Squares Regression

Table 1 indicates considerable variation between the three validation sets. For example RI ranges from MMRE=86% to MMRE=140%. Likewise the LSR
ranges from 38% to 100%. This is disappointing and indicates both approaches are sensitive to changes in the training set and may not cope well with heterogeneity. Second, we observe that the three accuracy indicators tend to favour the LSR approach over RI (Validation Set 3 is an exception). Third, we also observe a marked improvement when pruning the feature set for the RI method. It would seem that the algorithm does not deal effectively with categorical feature indicating the type of development environment. When DevEnv is removed there is striking improvement in the accuracy of RI prediction system so that is comparable to, or better than, the LSR method.

4. Discussion and Conclusions
The dataset used in this research had previously been used to test the effectiveness of effort estimation by analogy (Shepperd and Schofield 1997). Here an accuracy level of MMRE=64% was obtained, although in this case a jack knifing procedure was used across the entire dataset. Nevertheless we believe the results are broadly comparable.

Many software development environments consist of a complex set of interrelationships. Using the entire dataset without attempting to understand these factors can lead to substantially sub-optimal results. Preliminary results indicate that pruning the feature set can significantly improve the results from a rule induction approach. Heterogeneity within the dataset may also cause difficulties and there may be merit in partitioning the dataset, however, we have not yet explored this possibility.

Unfortunately the work carried out in this preliminary investigation has not tended to be very encouraging for use of rule induction methods to accurately predict software effort. Our experience of RI methods suggests that they can be unstable and predict less accurately than do other methods which in themselves are not regarded as good predictors. Having compared our results with a prediction system based upon a standard regression procedure, we conclude that rule induction does not offer a simple panacea to the problem of building software effort prediction systems. Nevertheless, we believe they warrant further investigation, particularly to try and explore under what conditions such approaches are most likely to be effective.

Acknowledgements
This research was partly funded by grant (Grant GR/L37298) from the UK Engineering and Physical Sciences Research Council and the Defence Research Agency. The authors are also grateful to Jean-Marc Desharnais for making his dataset available.
References